XDef 2019

对话 · 交流 · 合作　　前沿 · 实用 · 人才

第八届全国网络与信息安全防护峰会

动态沙箱检测与反检测技术进展

苏璞睿
中国科学院软件研究所
2019年12月13日

提 纲

- 一、背景及挑战
  - 动态沙箱是检测高级威胁的重要手段
- 二、环境敏感的恶意软件
  - 探测沙箱环境，对抗沙箱检测技术手段
- 三、环境敏感恶意软件的检测
  - 针对环境探测的各种检测思路及进展
- 四、我们的工作及进展
  - 金刚系统及相关进展

一、背景及挑战

高级持续性威胁（APT）攻击中，应用各种技术对抗当前的检测手段和防护策略，动态沙箱是检测高级威胁的重要手段，也是最有效的手段之一。

APT攻击检测是保障国家网络空间安全的重大战略需求



高级持续性威胁（APT）攻击是**有组织实施**、**高技术资源支撑**的**高隐蔽性**、**高破坏性攻击**，是当前网络空间对抗的最前沿，是国家重大战略需求

APT攻击检测防御是业界难题之一

技术对抗手段

反虚拟化 未知漏洞利用 专有控制协议 ROP

反沙箱对抗

类攻击代码 P2P僵尸网络 代码重用攻击 匿名网络

行为重组 JOP

代码变形 SMC自修改代码 环境探测与反制 Stalling Code迟滞代码

检测 → 分析 → 追踪

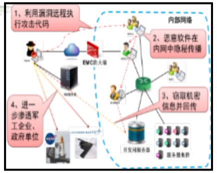
传统防御体系

入侵检测 杀毒软件 代码校验 反汇编

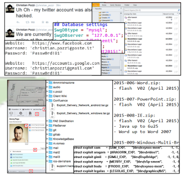
防火墙 反编译 流量特征检测

主动防御 网络隔离 程序调试

- 国际顶级安全公司RSA公司遭受攻击



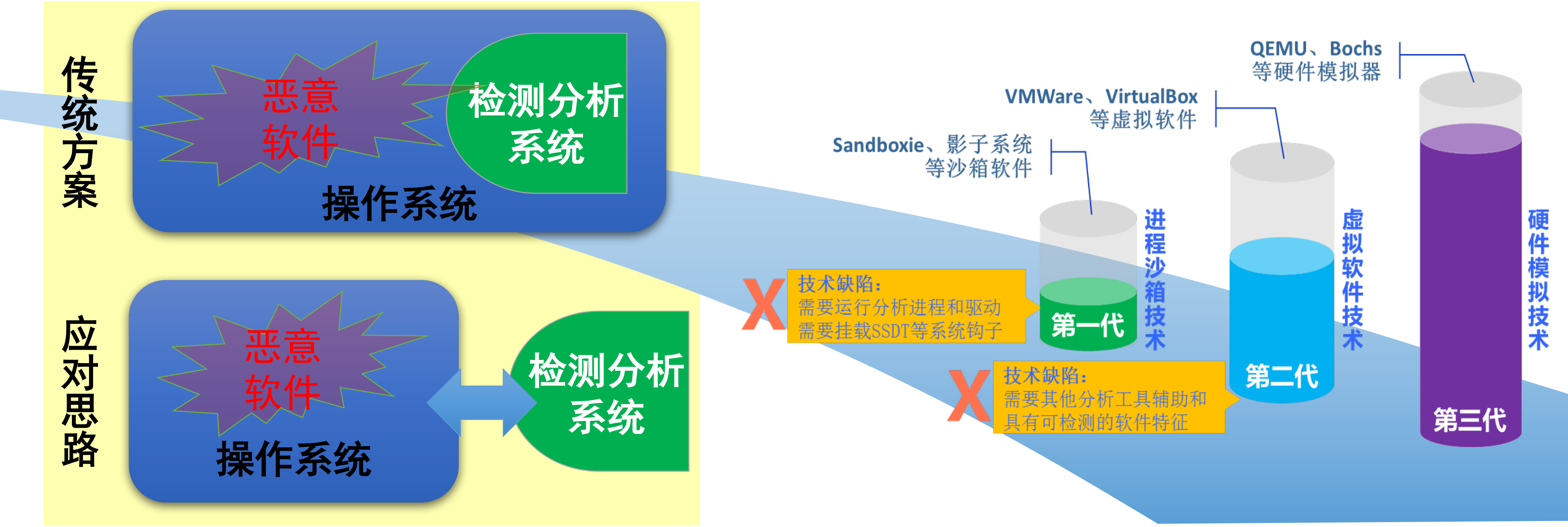
- 国际顶级攻击团队Hacking-Team遭受攻击



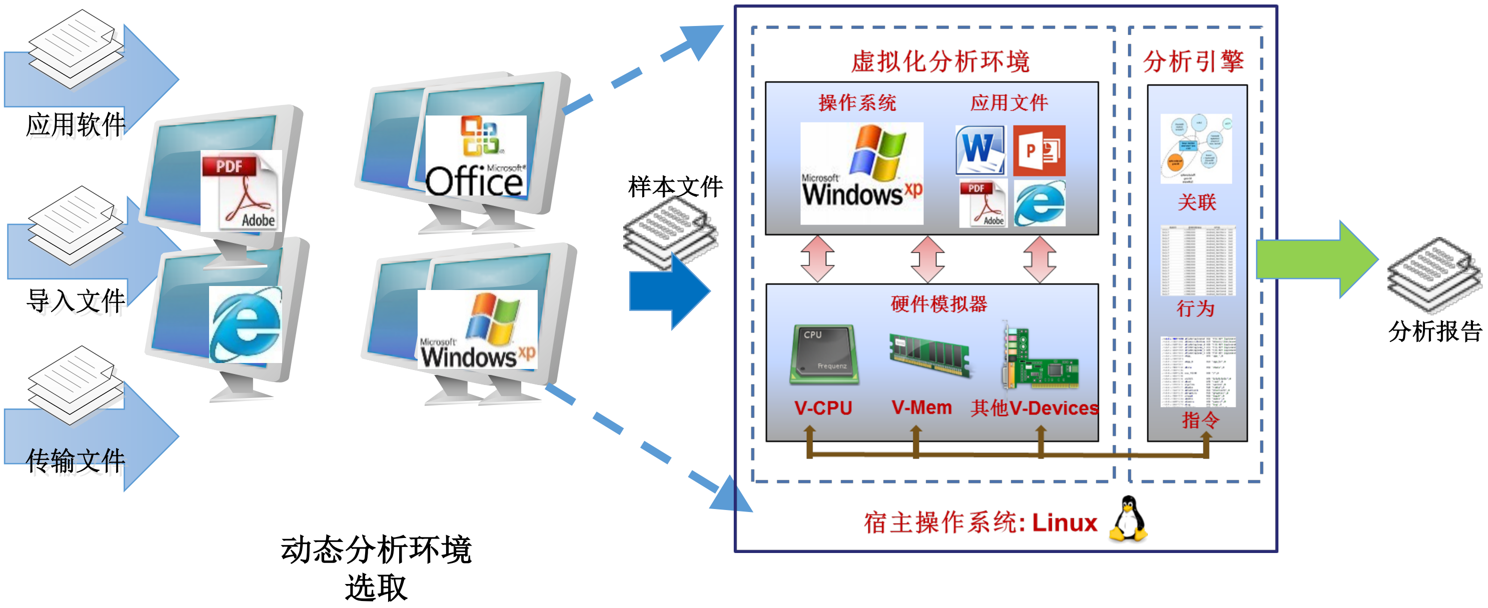
- 美国国家安全局NSA下属方程式组织遭受攻击



动态沙箱是检测APT攻击的重要手段之一



沙箱系统的典型工作模式（金刚系统）



动态沙箱环境的“真实性”是触发攻击的关键







二、环境敏感的恶意软件

沙箱的对抗与反对抗，A Never-ending Arms Race

高对抗APT攻击样本环境探测

高对抗背景下，攻击者采用各种手段规避动态检测

------ 样本行为触发不充分，机理分析困难

规避卡巴斯基的动态检测

```
v0 = CreateToolhelp32Snapshot()
while(Process32Next(v0, &pe))
   if(strnicmp(pe.szExeFile,"avp.exe",4)==0)
      return 1;
 }
return 0;
```

```
PROCESSENTRY32 pe; // [sp+8h] [bp-130h]@1

memset(&pe.cntUsage, 0, 0x124u);
pe.dwSize = 296;
v0 = CreateToolhelp32Snapshot(2u, 0);
if ( v0 == (HANDLE)-1 )
  return 0;
if ( !Process32Next(v0, &pe) )
{
LABEL_18:
  CloseHandle(v0);
  return 0;
}
while ( _strnicmp(&pe.szExeFile[1], "mtool", 5u)
     && _strnicmp(&pe.szExeFile[1], "llyd", 4u)
     && _strnicmp(&pe.szExeFile[1], "ython", 5u)
     && _strnicmp(&pe.szExeFile[1], "ilemo", 5u)
     && _strnicmp(&pe.szExeFile[1], "egmon", 5u)
     && _strnicmp(pe.szExeFile, "peid", 4u)
     && _strnicmp(&pe.szExeFile[1], "rocex", 5u)
     && _strnicmp(pe.szExeFile, "vbox", 4u)
     && _strnicmp(&pe.szExeFile[1], "iddler", 6u)
     && _strnicmp(&pe.szExeFile[1], "ortmo", 5u)
     && _strnicmp(&pe.szExeFile[1], "iresha", 6u)
     && _strnicmp(&pe.szExeFile[1], "rocmo", 5u)
     && _strnicmp(&pe.szExeFile[1], "utoru", 5u)
     && _strnicmp(&pe.szExeFile[1], "cpvie", 5u) )
{
  if ( !Process32Next(v0, &pe) )
    goto LABEL_18;
}
return 1;
}
```

在特定语言环境下才激活攻击行为

```
v0 = GetSystemUILanguage()
if(v0 == 0x419 || v0 == 0x422)
      return 1;
 }
return 0;
```

```
.text:004053E8                 dd 0C7E900E0h, 419B445h, 45C70000h, 422B8h, 0BC45C700h
.text:004053E8                 dd 423h, 28C045C7h, 0C7000004h, 42BC445h, 45C70000h, 42CC8h
.text:004053E8                 dd 0CC45C700h, 437h, 3FD045C7h, 0C7000004h, 440D445h, 45C70000h
.text:004053E8                 dd 442D8h, 0DC45C700h, 443h, 44E045C7h, 0C7000004h, 818E445h
.text:004053E8
.text:0040545C ; ---------------------------------------------------------------------------
.text:0040545C                 adc     eax, offset GetUserDefaultUILanguage
.text:00405461                 movzx   eax, ax
.text:00405464                 mov     [ebp-8], eax
.text:00405467                 call    ds:GetSystemDefaultUILanguage
.text:0040546D                 movzx   eax, ax
.text:00405470                 mov     [ebp-0Ch], eax
.text:00405473                 and     dword ptr [ebp-4], 0
.text:00405477                 jmp     short loc_405480
.text:00405479 ; ---------------------------------------------------------------------------
.text:00405479
.text:00405479 loc_405479:                             ; CODE XREF: .text:loc_4054A8↓j
.text:00405479                 mov     eax, [ebp-4]
.text:0040547C                 inc     eax
.text:0040547D                 mov     [ebp-4], eax
.text:00405480
.text:00405480 loc_405480:                             ; CODE XREF: .text:00405477↑j
.text:00405480                 cmp     dword ptr [ebp-4], 10h
.text:00405484                 jnb     short loc_4054AA
.text:00405486                 mov     eax, [ebp-4]
.text:00405489                 mov     eax, [ebp+eax*4-4Ch]
.text:0040548D                 cmp     eax, [ebp-8]
.text:00405490                 jnz     short loc_405497
.text:00405492                 xor     eax, eax
.text:00405494                 inc     eax
.text:00405495                 jmp     short loc_4054AC
```

对抗沙箱检测的主要思路

环境探测：通过环境特征判断是否是沙箱？

- 计算特征：动态沙箱需要构建虚拟环境以处理大规模样本（驱动、设备ID、CPU指令、特定服务……）
- 异常特征：沙箱需要全面监测数据，以实施检测判断（调试、指令插桩、SSDT劫持......）
- 高靶向性行为触发：仅针对特定目标触发行为

图灵测试：从用户行为特征判断是否目标用户？

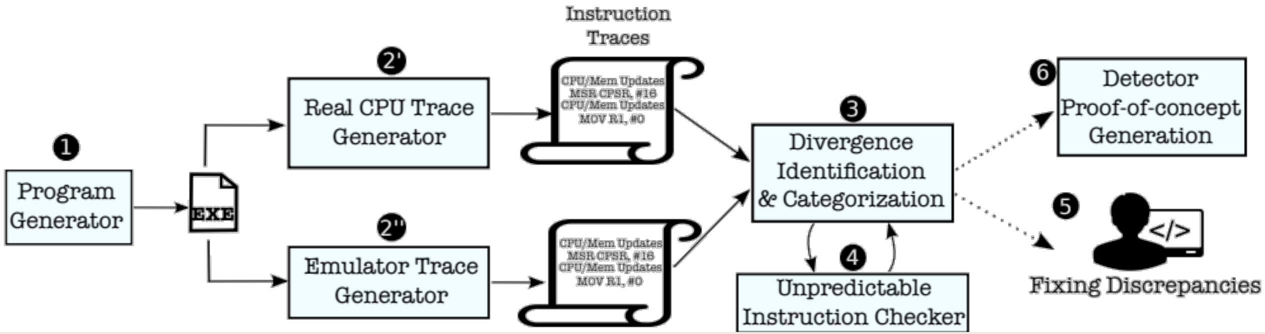
- 每个检测环境无法人为操作：鼠标等设备操作、网络访问、历史日志......

其他检测对抗方法

- DeepLocker：AI黑箱模型中嵌入检测对抗逻辑
- Stalling Code 迟滞代码（高强度干扰性数学运算、API Flood）
- Fileless 无文件攻击脚本（Powershell）

虚拟环境探测：以模糊测试确定指令仿真缺陷

- Proteus(RAID’ 18)：探测android虚拟沙箱环境
  ① 生成测试程序
  ② 提取真实CPU和沙箱模拟器指令记录
  ③ 识别指令执行差异：寄存器、内存和异常信息
  ④ 排除非确定类型指令的干扰



Sahin Onur,Coskun Ayse K.,Egele Manuel. Proteus: Detecting Android Emulators from Instruction-Level Profiles; In: Proc. of the Research in Attacks, Intrusions, and Defenses, Cham, F, 2018 [C]. Springer International Publishing.

虚拟环境探测：以模糊测试确定指令仿真缺陷

模拟器
VS
真实CPU

Fast Model Behavior	QEMU Response: SIGILL	SIGSEGV	None
und	149436 (30%)	826 (0.2%)	9341 (1.9%)
abt	11 (0.002%)	12471 (2.5%)	528 (0.1%)
svc	0	18932 (3.8%)	14609 (2.9%)
usr	5270 (1.1%)	176975 (35%)	0

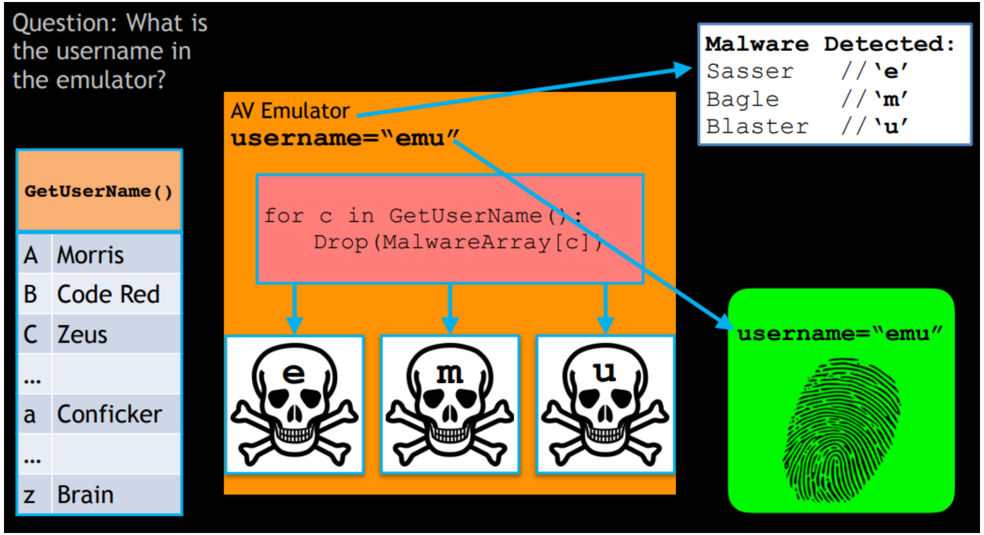
<und, !SIGILL>	10167 (2%)
<usr, SIGILL>	5270 (1.1%)
<abt, !SIGBUS>	13010 (2.6%)
<usr, SIGBUS>	0
mem_op_difference	200 (0.05%)
register_divergence	12 (0.002%)

真实CPU无效，但QEMU能够执行的指令

Instruction Encoding (cond $\in [0, 0xE]$)	Divergent Condition	QEMU Behavior	Real CPU Behavior	#Cases
cond:4|0001|op:4|*:12|1001|*:4	op = 1,2,3,5,6,7	SWP Inst.	Undefined	715
cond:4|1100010|*:9|101|*:1|op:4|*:4	op != 1,3	64-bit VMOV	Undefined	424
cond:4|11|op:6|*:20	op = 1,2	VFP Store	Undefined	51
cond:4|11101|*:2|0|*:8|1011|*:1|op:2|[illegible]	op = 2,3	VDUP Inst.	Undefined	3
cond:4|110|op:5|*:8|101|*:9	op != 4,5,8-25, 28,29	VFP Store	Undefined	2

通过黑盒测试自动抽取杀毒软件沙箱环境特征

① 标注恶意样本：每个样本分配特定ID
② 恶意样本加壳混淆：避免被静态检测
③ 将沙箱环境特征查询请求编码为执行逻辑，生成最终样本
④ 杀毒软件扫描样本，得到查杀结果
⑤ 依据查杀结果获取环境特征



Blackthorne Jeremy,Bulazel Alexei,Fasano Andrew,Biernat Patrick,Yener Bülent. AVLeak: fingerprinting antivirus emulators through black-box testing. Proceedings of the 10th USENIX Conference on Offensive Technologies. Austin, TX; USENIX Association. 2016: 91-105.

通过黑盒测试自动抽取杀毒软件沙箱环境特征

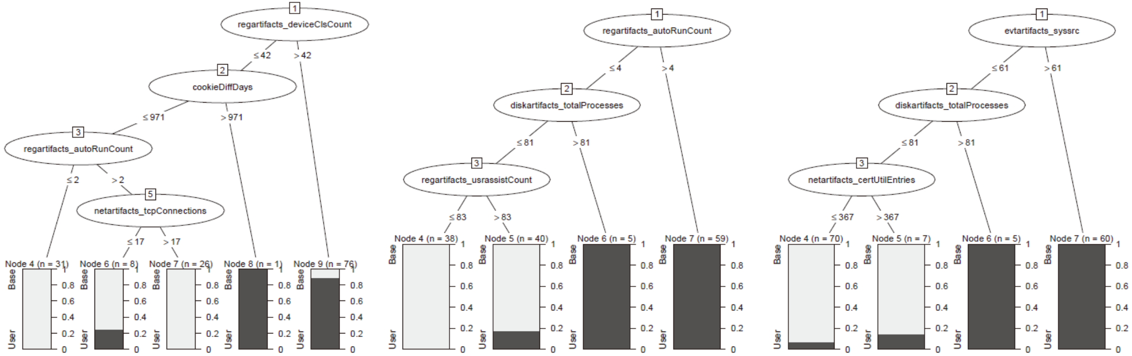
Avleak 可识别抽取杀毒软件沙箱中六大类特征

Class	Definition	Examples
Environmental Artifacts	Artifacts of the execution environment itself	Program / user / computer names, MAC addresses, file system, registry entries, running processes, GUI windows
OS API Inconsistency	Inconsistency in OS API implementation	Functions which return incorrect results, incorrectly fail or succeed
Network Emulation	Fingerprints related to network emulation	Hardcoded network responses, responses to invalid requests, protocol emulation inconsistencies
Timing	Timing inconsistencies related to emulation	Timing skews, inconsistencies, and failures
Process Introspection	Artifacts in the memory space of the program	Register states after API calls, runtime process structures, uninitialized memory, heap metadata, DLLs in memory
CPU "Red Pills"	Inconsistencies in CPU instruction emulation	Incorrectly emulated instructions, unique cycle counts, out-of-order execution behavior

Table 1: Table summarizing the six categories of fingerprints we examined with AVLeak.

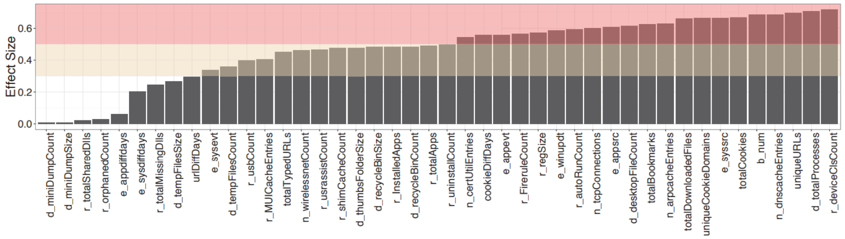
Spotless Sandboxes: Evading Malware Analysis System using Wear-and-Tear Artifacts

- 基于系统使用“磨损”痕迹，探测运行环境
  - 通过从真实用户设备和公开可用恶意软件分析服务收集使用磨损痕迹；
  - 基于使用年限和磨损程度构建统计分类模型
    - 收集的数据包括：系统信息类、磁盘信息、网络数据信息、注册表信息、浏览器痕迹
  - 基于决策树探测运行环境是否为检测系统



Miramirkhani N.,Appini M. P.,Nikiforakis N.,Polychronakis M. Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts; In: Proc. of the IEEE Symposium on Security and Privacy (SP), F, 2017 [C]

沙箱和真实用户机器“磨损”痕迹的差异



基于各类磨损信息的环境探测准确率
准确率达到92.86%以上

TABLE VI: Classifier performance when trained only on a single type of artifacts (FNR = False Negative Rate, FPR = False Positive Rate).

Artifact Type	#Artifacts	Accuracy	FNR	FPR
Network	4	95.92%	2.04%	6.12%
Disk	5	95.92%	0.00%	8.16%
System	5	92.86%	0.00%	14.29%
Registry	11	93.88%	4.08%	8.16%
Browser	7	92.86%	2.04%	12.24%

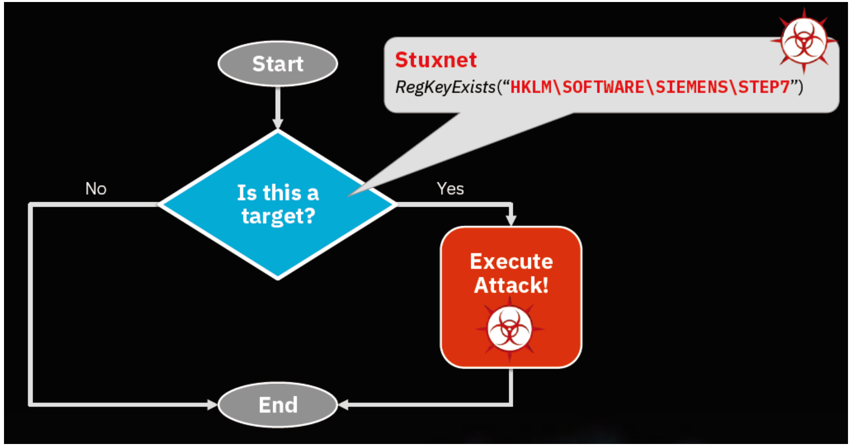
识别沙箱的用户实用痕迹类别

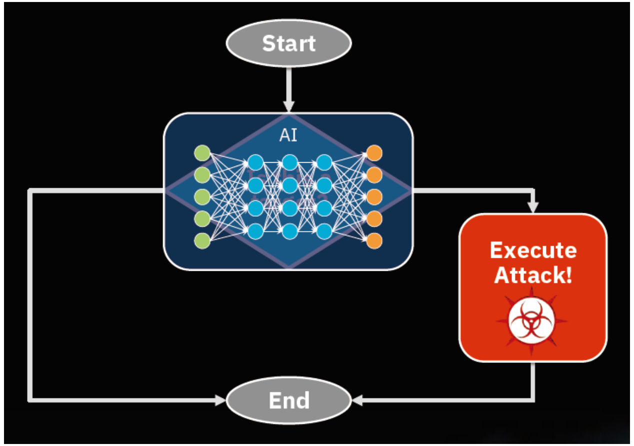
TABLE I: Complete list of wear-and-tear artifacts.

Category	Name	Description	User	Sandbox	Baseline
System	totalProcesses	# of processes	94.4	35	41
	winupdt	# installed Windows updates	794	19	2
	sysevt	# system of system events	27K	8K	334
	appevt	# of application events	18K	2.4K	184
	syssrc	# sources of system events	78	49	48
	appsrc	# sources of application events	40	26	23
	sysdiffdays	Elapsed time since the first system event (days)	370	1.7K	0
	appdiffdays	Elapsed time since the first application event (days)	298	943	0
Disk	recycleBinSize	Total size of the recycle bin (bytes)	2.5G	50M	0
	recycleBinCount	# files in the recycle bin	109	3	0
	tempFilesSize	Total size of temporary system files (bytes)	302	24	8.2
	tempFilesCount	# temporary system files	411	60	10
	miniDumpSize	Total size of process crash minidump files (bytes)	3M	409K	0
	miniDumpCount	# of process crash minidump files	9	4	0
	thumbsFolderSize	Total size of the system's thumbnails folder (bytes)	63M	8M	2.6M
	desktopFileCount	# files on the desktop	34	6	3
Network	ARPCacheEntries	# entries in the ARP cache	19	4.5	5
	dnscacheEntries	# entries in the DNS resolver cache	151	4	3
	certUtilEntries	# URLs of previously downloaded CRLs	1.7K	210	6
	wirelessnetCount	# of cached wireless SSIDs	8	0	0
	tcpConnections	# of active TCP connections	77	27	16
Registry	regSize	Size of the registry (in bytes)	144.8M	53M	35M
	uninstallCount	# registered software uninstallers	177	58	18
	autoRunCount	# programs that automatically run at system startup	9	3	1
	totalSharedDlls	Legacy DLL reference count	242	176	26
	totalAppPaths	# registered application paths	54	35	23
	totalActiveSetup	# Active Setup application entries	24	32	25
	orphanedCount	# leftover registry entries	11	10	9
	totalMissingDlls	# registered DLLs that do not exist on disk	21	23	13
	usrassistCount	# of entries in the UserAssist cache (frequently opened applications)	222	98	11
	shimCacheCount	# entries in the Application Compatibility Infrastructure (Shim) cache	191K	98K	38K
	MUICacheEntries	# Multi User Interface (MUI) cache entries	66	83	163
	FireruleCount	# of rules in the Windows Firewall	511	332	358
	deviceClsCount	# previously connected USB devices (DeviceInstance IDs)	81	29	60
	USBStorCount	# previously connected USB storage devices	1.7	0	0
Browser	browserNum	# installed browsers (Internet Explorer, Firefox, Chrome)	2.9	1.4	1
	uniqueURLs	# unique visited URLs	28K	13.8	1.64
	totalTypedURLs	# URLs typed in the browser's navigation bar	1.6K	4	1
	totalCookies	# of HTTP cookies	3K	135	2
	uniqueCookieDomains	# unique HTTP cookie domains	1003	23	1
	totalBookmarks	# bookmarks	274	20	1
	totalDownloadedFiles	# downloaded files	340	3	0
	urlDiffDays	Time elapsed between the oldest and newest visited URL (days)	225	370	0
	cookieDiffDays	Time elapsed between the oldest and newest HTTP cookie (days)	303	256	0

DeepLocker

- 在AI黑箱中隐藏目标选择和检测对抗逻辑



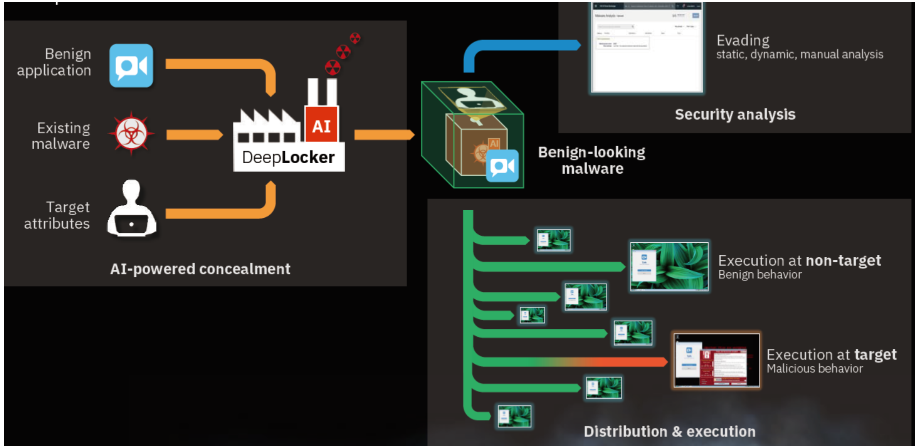


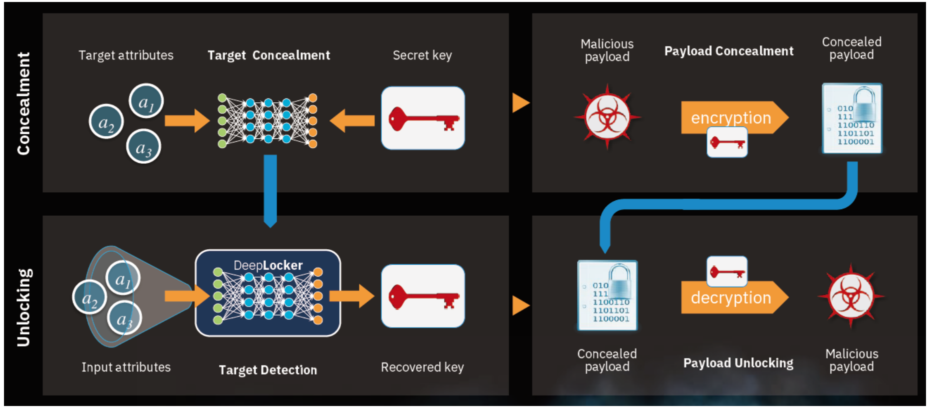
传统基于系统特征判断是否为攻击目标　DeepLocker基于AI算法判断是否为攻击目标

Dhilung Kirat, J.J., Marc Ph. Stoecklin, DeepLocker: Concealing Targeted Attacks with AI Locksmithing, in Blackhat USA. 2018, IBM Research

DeepLocker

- 采集外部环境信息
  - 收集目标物理环境、软件环境、用户行为、地理位置等属性信息
- 构建深度学习模型决策模型
  - 训练用于目标识别的多层神经网络模型，模型捆绑到恶意代码中用于目标环境判断
- 智能决策
  - 基于AI算法检测运行环境是否为目标环境；在非目标环境执行无害行为，在目标环境执行恶意行为





三、环境敏感恶意软件检测

如何构造高透明的检测环境，如何发现环境敏感的行为异常

沙箱反对抗的主要思路

高透明沙箱

- Bare-metal 沙箱：基于物理主机构建

规避行为检测

- 以Bare-meta沙箱为参照进行比对检测：BareCloud
- Stalling Code 迟滞代码检测

多路径执行

- 强制执行特定路径：X-force
- 执行环境主动构造：GoldenEye

BareBox: Efficient Malware Analysis on Bare-Metal

- 裸机硬件上进行恶意软件分析
- 无需重启的系统快速恢复
  - 内存数据备份恢复机制：
    - 内核实现Meta-OS，将物理内存重新分3个区，第一个用于TargetOS，第二个用于Target OS的备份，第三个用于Meta-OS,以支持内存数据的恢复
  - 基于RAM disk模拟磁盘驱动器，保障分析过程实际硬盘不被修改
- 软件监控和数据获取
  - 系统驱动：通过SSDT的监控记录恶意代码系统调用层面的行为

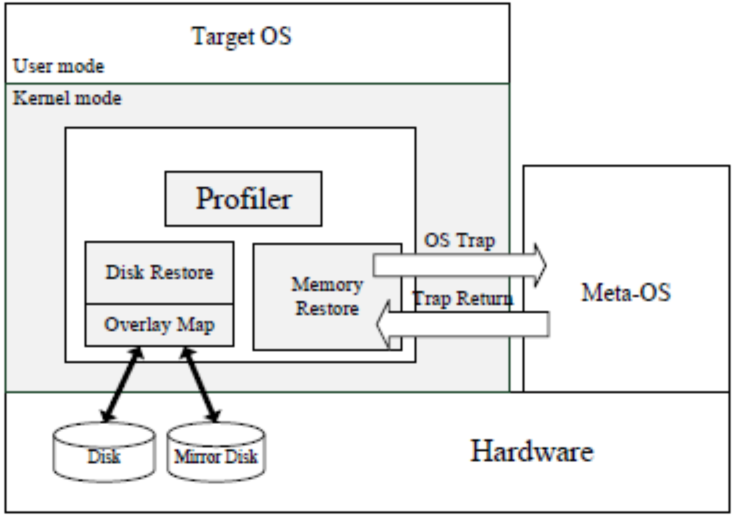
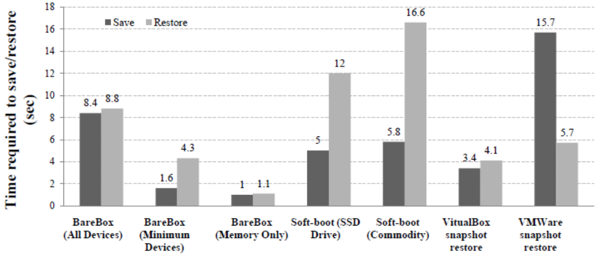


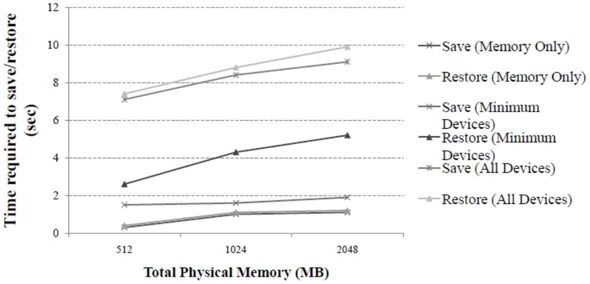
Figure 1: Architecture Overview

Kirat Dhilung,Vigna Giovanni,Kruegel Christopher. BareBox: efficient malware analysis on bare-metal. Proceedings of the 27th Annual Computer Security Applications Conference. Orlando, Florida, USA; ACM. 2011: 403-12.

BareBox: Efficient Malware Analysis on Bare-Metal

- 资源消耗比较





行为触发情况比较

Table 2: Number of new process creation

Malware Family	BareBox	VMware	QEMU
Rebhip	9	0	3
Telock	2	1	1
Conficker	0	0	0
Zlob/Zeus	10	10	4
Sdbot	4	1	1
Agobot	50	3	1
Armadillo-4	1	0	0

BareCloud:Bare-metal Analysis-based Evasive Malware Detection

- 检测恶意样本的“规避检测”行为
  - 不同环境下获取恶意样本的行为数据
    - Virtualization、Emulation、Hypervisor、Bare-metal环境下获取恶意样本的行为数据；
  - 检测其“规避检测”的行为
    - 比较样本在不同环境下行为的差异性
  - 将各个不同平台的行为结果按照统一规范进行表示

π =（O，R，P）,其中O是对象集合，R是操作集合，P $\subseteq (O \times R)$,

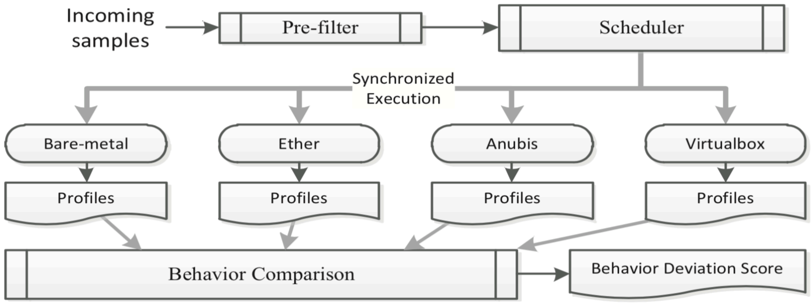


Figure 1: Overview of the system

$$O = (obj_type, obj_name)$$
$$obj_type ::= file|registry|sysconf|mbr|network$$

$$R ::= (op_name, op_attribute)$$

Kirat Dhilung,Vigna Giovanni,Kruegel Christopher. Barecloud: bare-metal analysis-based evasive malware detection. Proceedings of the 23rd USENIX conference on Security Symposium. San Diego, CA; USENIX Association. 2014: 287-301.

BareCloud:Bare-metal Analysis-based Evasive Malware Detection

- 采用层次化的相似度匹配方法
  - 层次结构与行为概要抽象级别相关联，避免因文件名等随机因素带来误差；

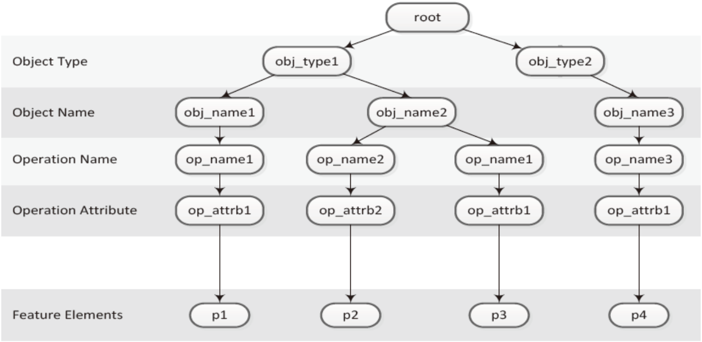


Figure 2: Behavior similarity hierarchy

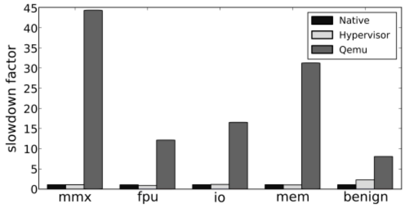
从110005个恶意样本中共发现了5835个具有“规避检测”行为的样本

Table 2: Evasion detection on different environments

Environment	Detection count	Percentage
Anubis	4,947	84.78
Ether	4,562	78.18
VirtualBox	3,576	61.28
All	2,530	43.35
Total	5,835	

迟滞代码(Stalling Code)检测

典型迟滞代码

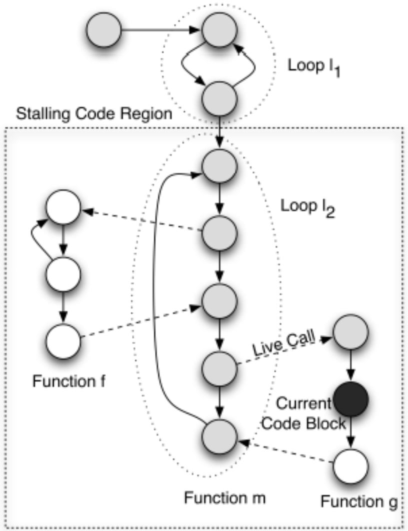


沙箱性能损失

```
unsigned count, t;

void helper() {
  t = GetTickCount();
  t++;
  t++;
  t = GetTickCount();
}
```

```
void delay() {
  count=0x1;
  do {
   helper();
   count++;
  } while
   (count!=0xe4e1c1);
}
```



迟滞代码
动态控制流图

① 动态插桩并构建动态控制流图
- 基本块开头
- Call和ret指令

② 识别活跃循环

③ 识别迟滞循环
- 启发式规则：最内层活跃循环
- 缓解性能损失方法：
  - 被动方法：减少监控点
  - 主动方法：主动终止迟滞代码

Kolbitsch Clemens,Kirda Engin,Kruegel Christopher. The power of procrastination: detection and mitigation of execution-stalling malicious code. Proceedings of the 18th ACM conference on Computer and communications security. Chicago, Illinois, USA; ACM. 2011: 285-96.

迟滞代码(Stalling Code)检测

最多超过66.5%的样本实现了迟滞代码绕过，触发22.3%的关键行为

Table 1: Additional behavior observed in Hasten (using *average improvement*).

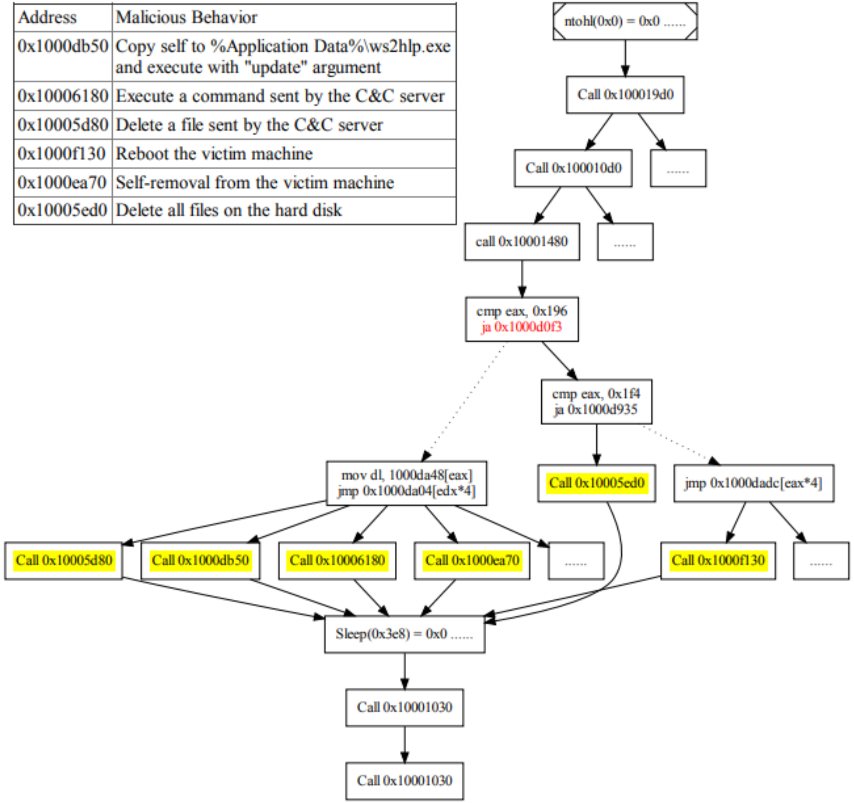
Description	Passive			Active		
	# samples	%	# AV families	# samples	%	# AV families
Runs total	**3,770**	–	319	**2,467**	–	231
Added behavior (any activity)	**1,003**	**26.6%**	119	**549**	**22.3%**	105
- Added file activity	949	25.2%	113	359	14.6%	79
- Added network activity	444	11.8%	52	108	4.4%	31
- Added GUI activity	24	0.6%	15	260	10.5%	51
- Added process activity	499	13.2%	55	90	3.6%	41
- Added registry activity	561	14.9%	82	184	7.5%	52
Ignored (possibly random) activity	**1,447**	**38.4%**	128	**276**	**11.2%**	72
No new behavior	**1,320**	**35.0%**	225	**1,642**	**66.5%**	174
- Exception cases	0	0.0%	0	277	11.2%	63

强制执行特定路径：X-force

牺牲精度换取更好路径覆盖能力

- 强制改写“污点”分支条件
  - 改写造成程序状态异常
    - 内存异常：按需分配内存
    - 输入缺失：实时提供随机值
  - 特殊情况处理
    - Switch跳转表，循环和递归
    - 栈和堆覆写保护
    - 外部库函数，多线程改为串行
- 多路径遍历策略
  - 路径覆盖优先
  - “间接跳转”覆盖优先

Address	Malicious Behavior
0x1000db50	Copy self to %Application Data%\ws2hlp.exe and execute with "update" argument
0x10006180	Execute a command sent by the C&C server
0x10005d80	Delete a file sent by the C&C server
0x1000f130	Reboot the victim machine
0x1000ea70	Self-removal from the victim machine
0x10005ed0	Delete all files on the hard disk



Peng Fei,Deng Zhui,Zhang Xiangyu,Xu Dongyan,Lin Zhiqiang,Su Zhendong. X-force: force-executing binary programs for security applications. Proceedings of the 23rd USENIX conference on Security Symposium. San Diego, CA; USENIX Association. 2014: 829-44.

强制执行特定路径：X-force

相比于传统沙箱，X-force在APT样本的动态执行中触发更多路径

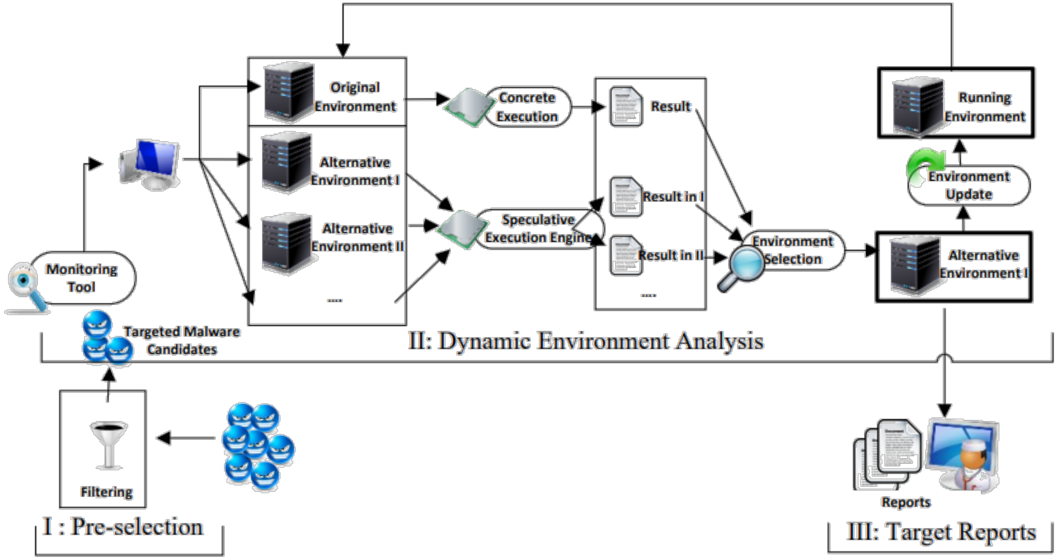
Name	MD5	File Size(KB)	Number of Library Call Sites		
			IDA Pro	Native Run	X-Force
dg003.exe	4ec0027bef4d7e1786a04d021fa8a67f	192	808	546	1750
Win32/PWSteal.F	04eb2e58a145462334f849791bc75d18	20	9	28	94
APT1.DAIRY	995442f722cc037885335340fc297ea0	19	213	68	236
APT1.GREENCAT	0c5e9f564115bfcbee66377a829de55f	14.5	303	114	302
APT1.HELAUTO	47e7f92419eb4b98ff4124c3ca11b738	8.5	109	33	109
APT1.STARSYPOUND	1f2eb7b090018d975e6d9b40868c94ca	7	80	15	80
APT1.WARP	36cd49ad631e99125a3bb2786e405cea	45.5	495	156	414
APT1.NEWSREEL	2c49f47c98203b110799ab622265f4ef	21	189	49	192
APT1.GOGGLES	57f98d16ac439a11012860f88db21831	10.5	127	45	131
APT1.BOUNCER	6ebd05a02459d3b22a9d4a79b8626bf1	56	24	39	562

通过渐进式预执行选择沙箱环境：GoldenEye

① 环境敏感样本筛选
- 污点分析

② 基于环境特征动态构建多个执行环境
- 穷举环境特征所有可能值
  - 语言：中文、英文等

③ 沙箱环境选取：污点辅助的并发预执行
- 渐进式分析：每次选取一个基本块，多个环境同时执行
- 启发式优化策略
  - 包含创建进程、线程等敏感操作
  - 没有调用Exit等退出函数



Xu Zhaoyan,Zhang Jialong,Gu Guofei,Lin Zhiqiang. GoldenEye: Efficiently and Effectively Unveiling Malware’s Targeted Environment. Research in Attacks, Intrusions and Defenses. Cham; Springer International Publishing. 2014: 22-45.

通过渐进式预执行选择沙箱环境：GoldenEye

针对Conficker等恶意软件样本，实现了系统、网络、硬件、进程等环境特征的识别与环境智能选择

	System	Network	Hardware	Customized Object	Library Process
Conficker[43]	√	√		√	√
Zeus[21]	√	√	√	√	√
Sality[12]	√	√			
Bifrost[2]	√	√	√	√	
iBank[7]	√	√	√	√	√
nuclearRAT[9]	√	√	√	√	√
Duqu[4]	√	√	√	√	√
Nitro[16]	√	√	√		
Qakbot[11]	√			√	√

(a) Ground Truth

	System	Network	Hardware	Customized Object	Library Process
Conficker	√	√		○	√
Zeus	√	√	√	√	○
Sality	√	√			
Bifrost	√	√	√	○	
iBank	√	√	×	×	√
nuclearRAT	√	×	√	○	○
Duqu	○	√	√	√	√
Nitro	○	√	○		
Qakbot	○			√	√

(b) GOLDENEYE Environment Extraction Result

√: Correctly Extracted, ○: Similar Element

×: Not Extracted

PHP恶意软件的多路径分析：MalMax

❶ 反事实（Counterfactual）执行

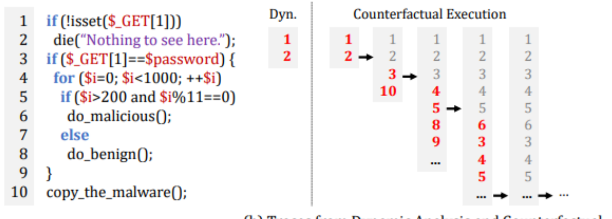
- 路径分支条件不满足
- 非正常执行路径
- 动态解析语句(eval)

❷ 控制流削减

- 基于循环识别的强制改写
- 基于覆盖率的循环次数阈值动态调整

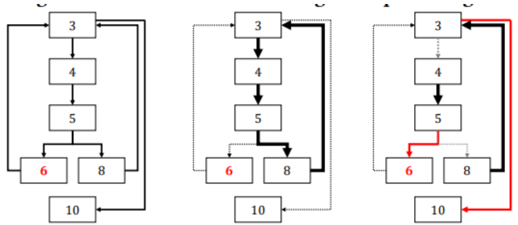
❸ 协作式单独路径执行

- 资源共享：全局变量、数据库连接、程序配置



(a) Source code

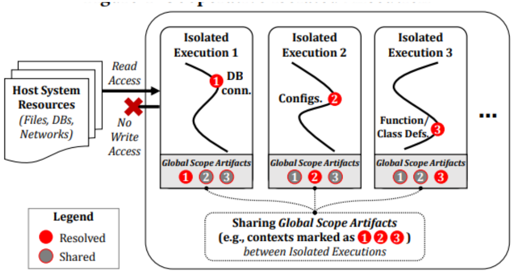
(b) Traces from Dynamic Analysis and Counterfactual Exec. (Numbers in traces are line numbers and an arrow means a creation of a new isolated execution)



(a) Control Flow Graph (CFG)

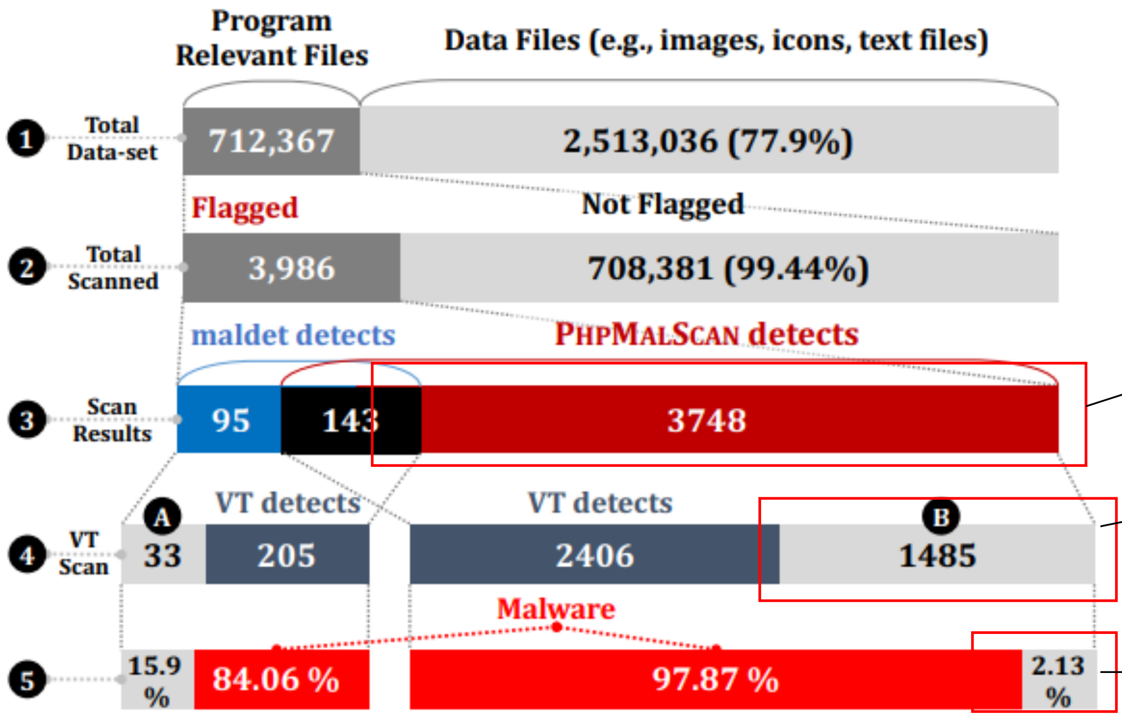
(b) Weighted CFG (after 100 iteration)

(c) Trimmed CFG (Help exploring the red paths)



Naderi-Afooshteh Abbas,Kwon Yonghwi,Nguyen-Tuong Anh,Razmjoo-Qalaei Ali,Zamiri-Gourabi Mohammad-Reza,Davidson Jack W. MalMax: Multi-Aspect Execution for Automated Dynamic Web Server Malware Analysis. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. London, United Kingdom; ACM. 2019: 1849-66.

322万文件（超过1TB）的检测结果 -- MalMax



参与对比的多个引擎共检出3986个恶意样本

相比于maldet，额外检出3748个恶意样本

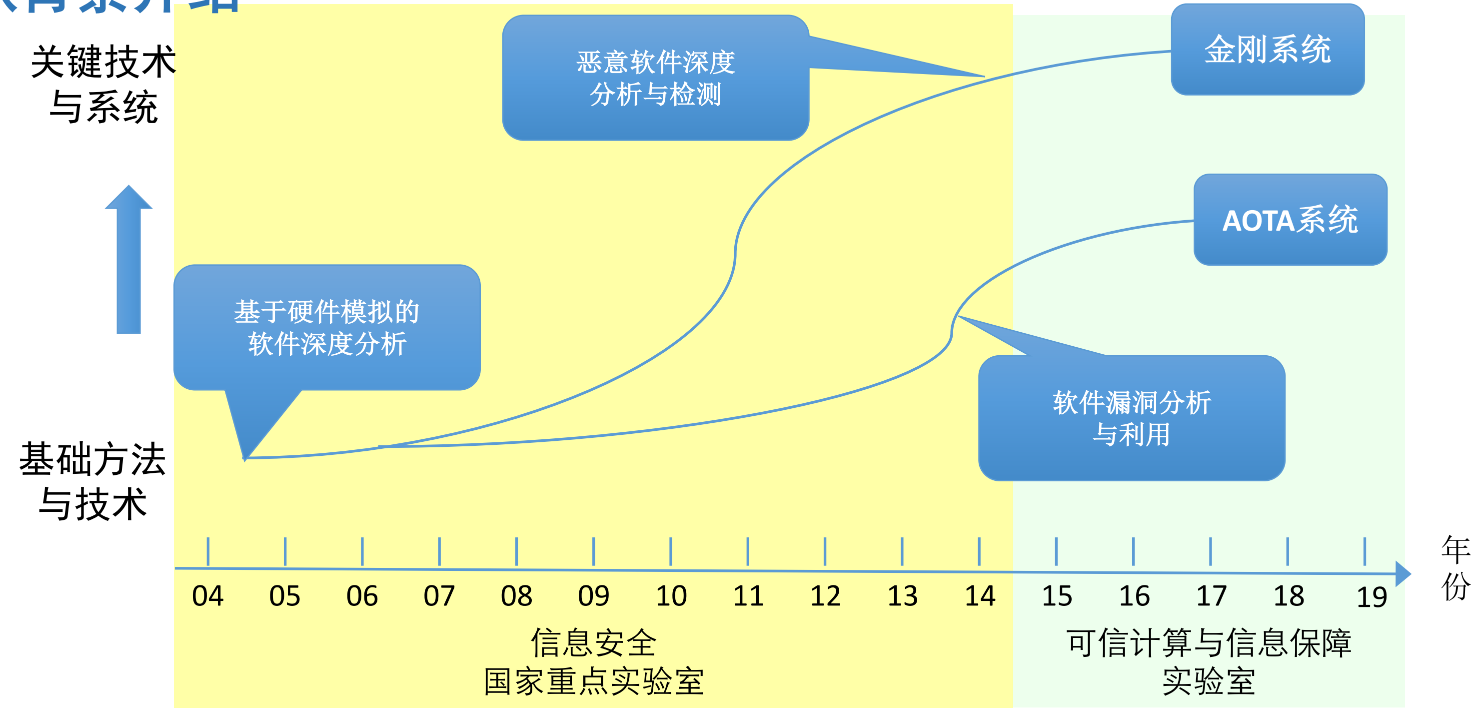
相比于VT，额外检出1485个恶意样本

误报率仅为2.13%

四、我们的研究进展

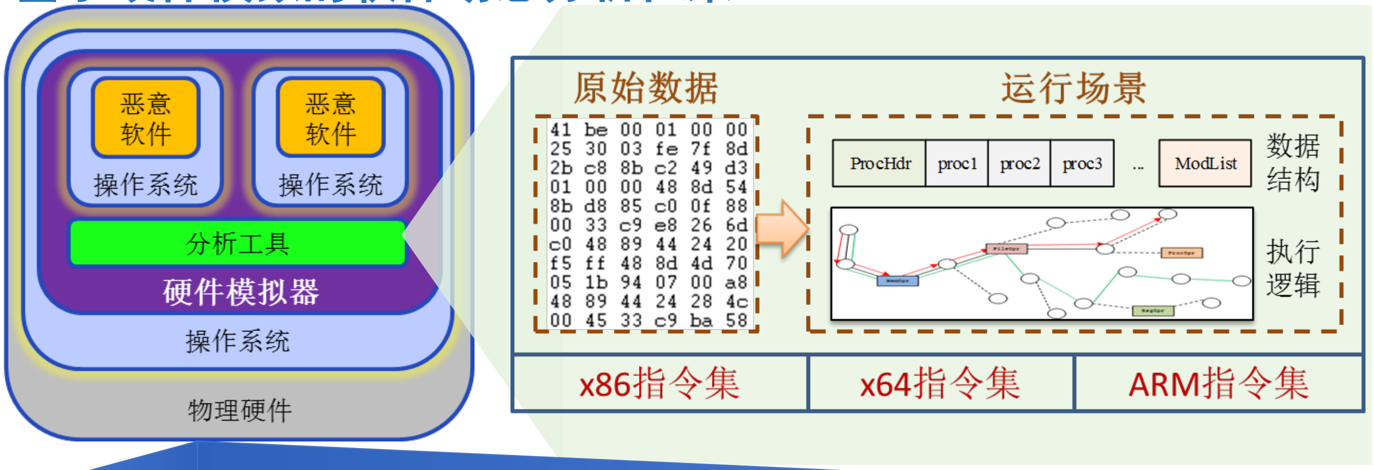
金刚系统及相关进展

团队背景介绍



研究工作得到了国家自然科学基金、国家863计划、国家科技支撑计划、国家信息安全产业化专项、国家重点研发计划项目等项目的持续支持！

关键技术：基于硬件模拟的软件动态分析框架



实现了**硬件层、指令级**的动态分析能力

- **细粒度**：对CPU指令等操作实时监控与内容解析
- **高透明**：基于软件运行过程场景恢复的动态分析
- **防对抗**：实现了时钟加速等防对抗措施
- **高性能**：分析工具仅额外增加0.12%的CPU资源

核心专利：ZL200810117899.X、ZL201110278249.5、ZL200810241105.0、ZL200910081510.5、ZL201110029135.7

基于硬件模拟的软件动态分析与检测

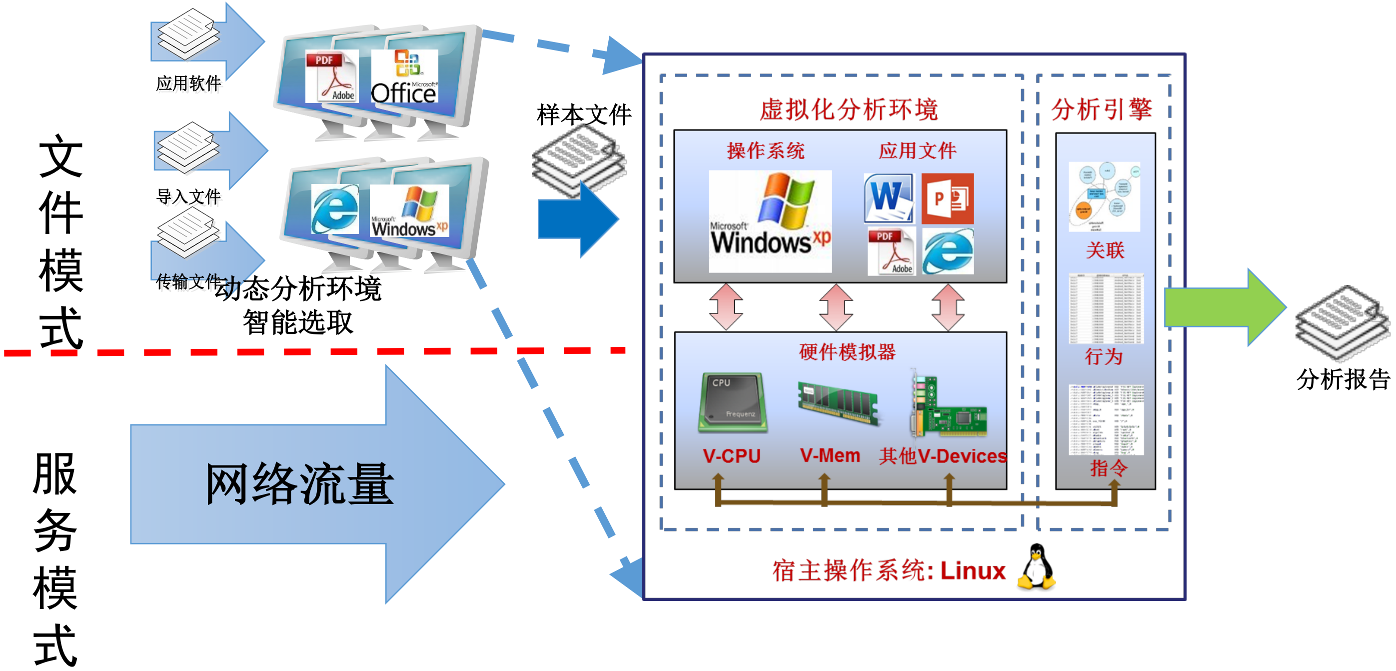
攻击检测

- 利用硬件模拟中的指令级监控能力，提升漏洞利用攻击的检测能力和检测准确性

机理分析

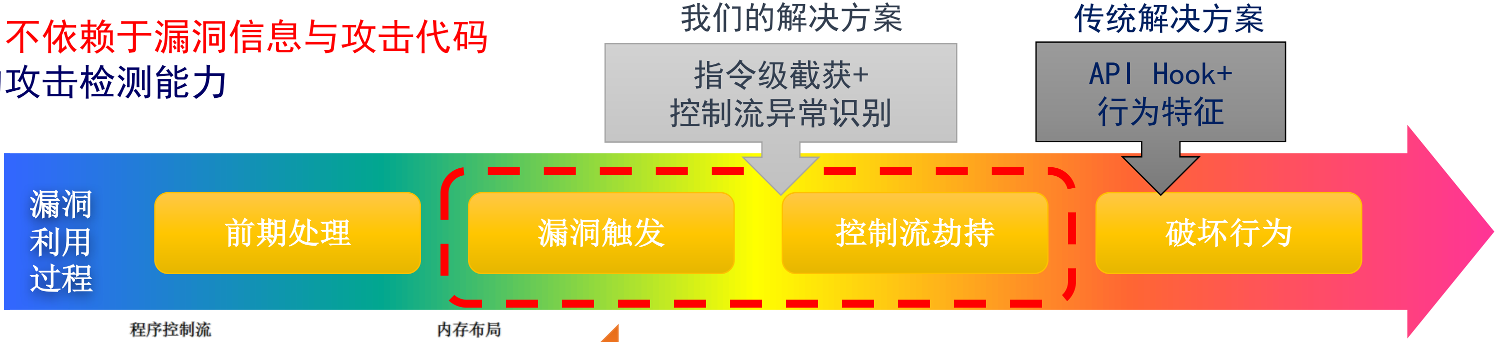
- 主要利用硬件模拟中的环境操控能力，结合符号执行等技术，提升环境构造能力，从而提高攻击行为触发能力

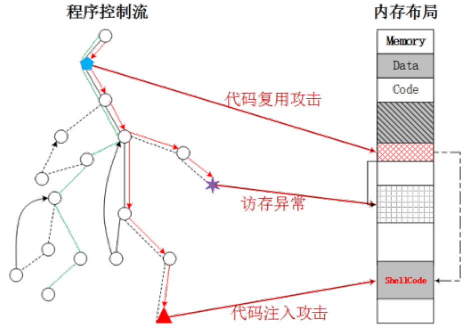
攻击检测：金刚系统


文件模式
应用软件
导入文件
传输文件
动态分析环境
智能选取
样本文件
虚拟化分析环境
操作系统
应用文件
硬件模拟器
V-CPU
V-Mem
其他V-Devices
分析引擎
关联
行为
指令
宿主操作系统: Linux
分析报告
服务模式
网络流量

关键技术：基于指令级异常的未知恶意软件检测方法

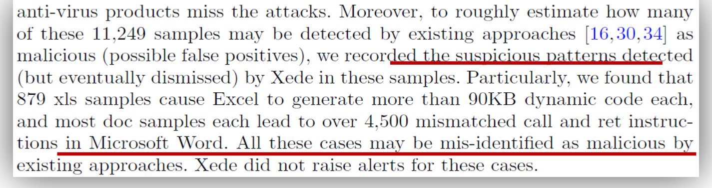
- 形成了不依赖于漏洞信息与攻击代码特征的攻击检测能力





异常控制流检测算法

时机前移，粒度更细，检测能力提升

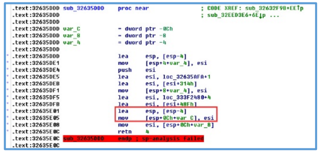

anti-virus products miss the attacks. Moreover, to roughly estimate how many of these 11,249 samples may be detected by existing approaches [16,30,34] as malicious (possible false positives), we recorded the suspicious patterns detected (but eventually dismissed) by Xede in these samples. Particularly, we found that 879 xls samples cause Excel to generate more than 90KB dynamic code each, and most doc samples each lead to over 4,500 mismatched call and ret instructions in Microsoft Word. All these cases may be mis-identified as malicious by existing approaches. Xede did not raise alerts for these cases.

- Meining Nie, Purui Su, Qi Li, Zhi Wang, Lingyun Ying, Jinlong Hu, Dengguo Feng. Xede: Practical Exploit Early Detection. The 18th International Symposium on Research in Attacks, Intrusions and Defenses. RAID 2015, Springer, 2015: 198-221.

代码复用检测方法:分析call-ret配对，检测ROP攻击

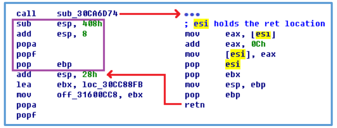
- 改进影子栈，应对现代软件广泛存在的“类攻击代码”：

  - 改进策略：以连续N个不匹配call-ret判定ROP（本文设N=3）



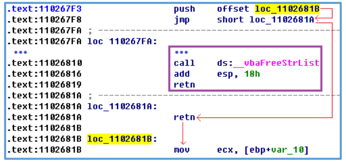
类攻击代码每次替换一个返回地址，而ROP攻击通常需要在17~30个gadgets之间跳转

  - 改进策略：采用模糊匹配算法，容错M字节（本文设M=16）



类攻击代码对返回地址的推移通常在几个字节以内，而ROP攻击的gadgets之间距离很大

  - 改进策略：自顶向下搜索，弹出已匹配元素之上的所有元素



应对setjmp/longjmp、call/pop和异常处理机制等高级编程技巧带来的有call无ret问题

有效性评估

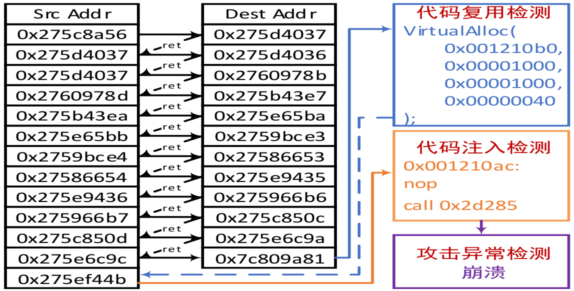
收集恶意代码检测

- 从exploit-db等处采集**12501**例doc、pdf等数据类型恶意样本

攻击技术	样本比例
代码注入攻击	75%
ROP攻击	51.47%
JOP攻击	19.85%
攻击失败引发异常	25%

漏洞利用案例分析

- CVE-2012-0158（成功攻击）



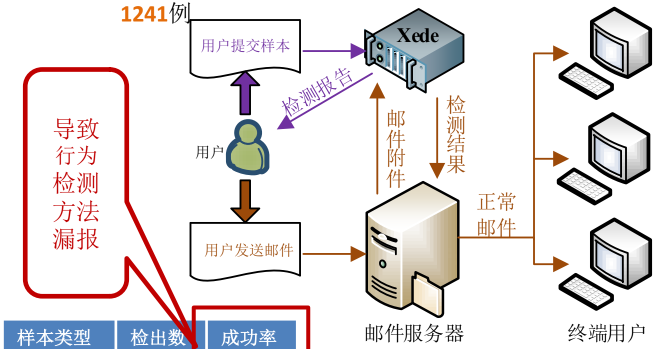
- CVE-2014-1761（失败攻击）

访存异常发生在0x909092e4

实际部署应用

实际部署应用效果

- 为互联网公众提供样本在线检测服务，三个月收集样本**1241**例
- 为某高校提供邮件附件随机抽检服务，收集样本**10144**例



样本攻击成功率

样本类型	检出数	成功率
doc/docx	58	74.14%
pdf	42	9.52%
swf	8	0%
xls/xlsx	11	72.73%
htm/html	6	50%
rtf	6	16.67%
ppt/pps	4	25%
wps	1	0%

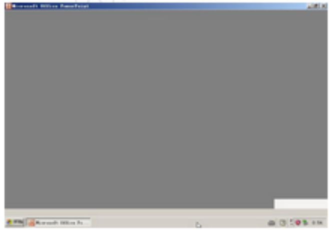
对比杀毒软件的漏报率

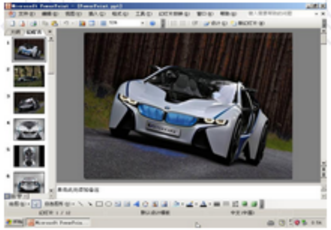
杀毒软件	更新日期	漏报率
Kaspersky	24/05/2015	21/136
McAfee	24/05/2015	49/136
Avira	24/05/2015	22/136
Norton	24/05/2015	32/136

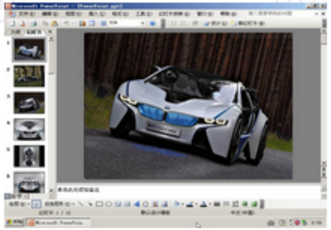
检测分析结果

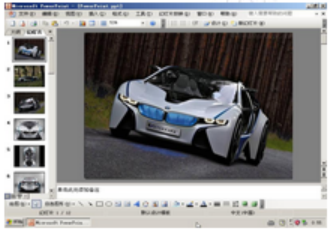
- 案例：PPT文件利用Office漏洞，可执行任意攻击代码，由于当时漏洞未公开，攻击过程可绕过最新各类杀毒软件检测

运行过程截图


截图 1


截图 2


截图 3


截图 4

检测分析结果

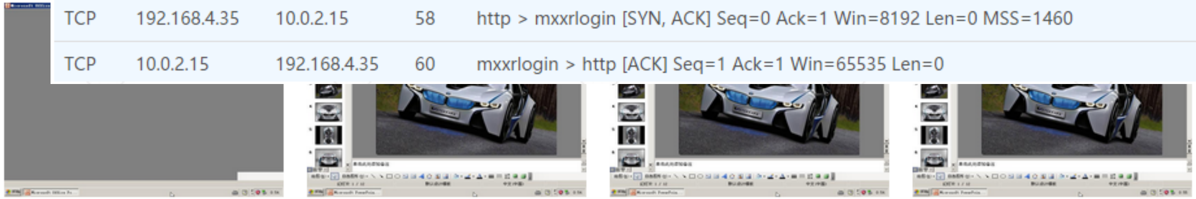
检测结果报告

恶意行为类目

行为逻辑关系图

典型联网活动追踪 联网活动下载

协议	源地址	目标地址	长度	网络数据
DNS	10.0.2.15	10.0.2.3	80	Standard query A downs.indian-info.in
DNS	10.0.2.3	10.0.2.15	96	Standard query response A 192.168.4.35
TCP	10.0.2.15	192.168.4.35	62	activesync > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
TCP	192.168.4.35	10.0.2.15	58	http > activesync [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
TCP	10.0.2.15	192.168.4.35	60	activesync > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
HTTP	10.0.2.15	192.168.4.35	178	GET /httpdocs/mm/xptest-f448c8c9:52-54-00-12-34-56/Cmwhite HTTP/1.1
TCP	192.168.4.35	10.0.2.15	54	http > activesync [ACK] Seq=1 Ack=125 Win=8760 Len=0
TCP	10.0.2.15	192.168.4.35	62	mxxrlogin > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
TCP	192.168.4.35	10.0.2.15	58	http > mxxrlogin [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
TCP	10.0.2.15	192.168.4.35	60	mxxrlogin > http [ACK] Seq=1 Ack=1 Win=65535 Len=0



截图 1 截图 2 截图 3 截图 4

该成果已规模化应用，并获得多项荣誉

APEC CHINA 2014

BEIJING 2015

数博会 BIG DATA EXPO 中国·贵阳 GUIYANG·CHINA

70 1945-2015

G20 2016 CHINA

2016 年二十国集团中国峰会



荣誉证书

北京市科学技术奖

为表彰在推动科学技术进步、对首都经济建设和社会发展作出贡献的集体和个人，特颁此证，以资鼓励。

获奖项目：恶意软件深度分析与检测关键技术及应用

获奖等级：贰等奖

获奖单位：中国科学院软件研究所、北京信息安全测评中心、国家计算机网络与信息安全管理中心、清华大学、中兴通讯股份有限公司、成都科来软件有限公司

北京市人民政府

二〇一七年十一月

NO. 2017 计-2-002

2017年北京市科学技术二等奖



中国通信学会科学技术奖

荣誉证书

获奖项目：APT攻击检测关键技术研究与应用

获奖等级：一等

获 奖 者：苏璞睿（第一完成人）

2018年12月

证 书 号：KJ2018-1-11-G01

2018年中国通信学会一等奖

基于执行环境主动构造的恶意软件多路径分析

指令级分析检测规避行为
抽取规避特征

主动构造执行环境
挖掘攻击样本隐蔽行为

☞反沙箱规避行为探测

通过污点分析检测影响控制流的环境探测行为（指令）

☞规避行为特征抽取

环境探测参数作为符号，基于混合符号执行技术抽取行为依赖条件

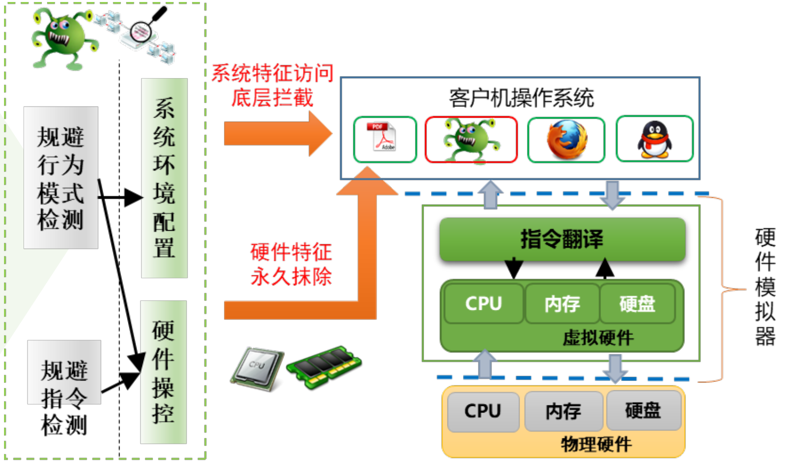
☞特征规范化和化简

依据规避行为语义和执行上下文对特征进行语义标识和条件化简

```
v0 = CreateToolhelp32Snapshot()
while(Process32Next(v0, &pe))
  if(strnicmp(pe.szExeFile,"avp.exe",4)==0)
    return 1;
}
return 0;
```

Str.Equal(CurrentProcess.name, “avp.exe”)

禁止特定进程运行（“avp.exe”）



支持15种规避行为的分析检测和执行环境构造

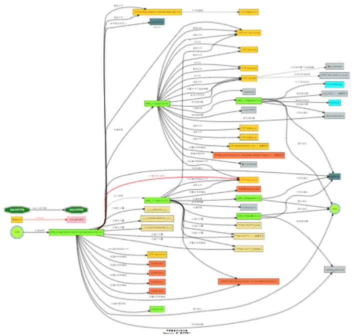
序号	规避行为	规避行为条件表达式
1	指定目录是否存在	DIRECTORY_EXIST("C:\\Program Files\\Oracle\\VirtualBox Guest Additions\\")
2	磁盘类型检测	MEDIA_DRIVE_TYPE(C) != 2
3	指定文件是否存在	Not FILE_EXIST(\"\\\\.\\VBoxMiniRdrDN\")
4	内存大小检测	ULT(MEMORY_SIZE, 1073741824)
5	互斥量存在检测	Not MUTEX_EXIST(\"oie20hgioedo\")
6	指定注册表键是否存在	Not REG_KEY_EXIST(HKEY_LOCAL_MACHINE\\SOFTWARE\\Oracle\\VirtualBox Guest Additions)
7	系统界面语言探测	And(UI_LANGUAGE != 1049, UI_LANGUAGE != 1058, UI_LANGUAGE != 1059, UI_LANGUAGE != 1064, UI_LANGUAGE != 1067, UI_LANGUAGE != 1068, UI_LANGUAGE != 1079, UI_LANGUAGE != 1087)
8	指定URL是否可访问	URL_OPEN(\"http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com\")
9	指定标题窗口是否存在	Not WINDOW_TITLE_EXIST(\"VBoxTrayToolWnd\")
10	指定名称进程是否存在	Not PROCESS_EXIST("rtvscan.exe")
11	当前样本路径检测	SAMPLE_FILEPATH_IS("C:\WINDOWS\system32\systray32.exe")
12	注册表键值内容是否包含某特定字符串	Not REGISTRY_KEYVALUE_EXIST(\"HKEY_CURRENT_USER\\Keyboard Layout\\Preload\\2=00000415\")
13	当前用户是否是指定用户名	Not USER_NAME_CONTAIN("SANDBOX")
14	计算机主机名检测	Not COMPUTER_NAME_IS("Sandbox")
15	主机CPU核数检测	ULT(CPU核数, 2)

某恶意软件的执行环境构造实例

---------- 依据不同名字互斥量的创建情况分别实施不同的行为

原始报告行为

	创建可执行程序	[高危]	创建可执行程序 HarddiskVolume1\dir\install\install\taskhost.exe
	自我复制	[高危]	复制 C:\vastest\b5085d1c9b420e3c4936ab16b66c2d02.exe 到 c:\dir\install\install\taskhost.exe，实现感染
+	设置自启动	[高危]	将 Policies 设置为开机自启动
+	设置注册表中键值	[敏感]	设置注册表中键 \REGISTRY\MACHINE\SOFTWARE\MICROSOFT\ACTIVE SETUP\INSTALLED COMPONENTS\{858706I6-K7L8-4W1Y-63AY-8LM25P3O6UKX} 的值 StubPath
+	写入文件	[高危]	写入文件 HarddiskVolume1\DOCUME~1\xpuser\LOCALS~1\Temp\XX--XX--XX.txt
	跨进程内存写入	[高危]	修改进程 explorer.exe 的内存数据
	创建远程线程	[高危]	注入 explorer.exe 进程
+	读取文件	[敏感]	读取文件 HarddiskVolume1\DOCUME~1\xpuser\LOCALS~1\Temp\XX--XX--XX.txt
+	映射文件	[普通]	映射文件 HarddiskVolume1\WINDOWS\explorer.exe
+	创建进程	[高危]	创建进程 explorer.exe
+	检测分析环境	[普通]	通过 CpuCheck 技术检测虚拟机或调试器，逃避分析与查杀
	变更权限	[高危]	改变进程的权限
+	删除文件	[敏感]	删除文件 C:\DOCUME~1\xpuser\LOCALS~1\Temp\XX--XX--XX.txt
	修改文件时间	[高危]	修改 HarddiskVolume1\Documents and Settings\xpuser\Application Data\logs.dat 的创建时间为 2005-04-05 02_16_43
	修改文件属性	[高危]	将文件 HarddiskVolume1\Documents and Settings\xpuser\Application Data\logs.dat 的属性更改为隐藏或只读
	读取系统文件	[敏感]	读取文件 HarddiskVolume1\WINDOWS\Registration\R000000000007.clb
	查询系统服务	[高危]	查询系统服务
	控制系统服务	[高危]	控制 ProtectedStorage 服务，执行恶意操作
	系统调用异常	[敏感]	异常的系统调用方式



创建互斥量

___PasswordList___

环境探测行为

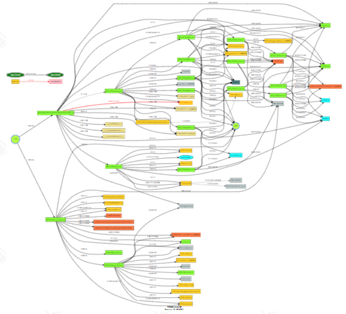
	进程	行为信息概要
	b5085d1c9b420e3 : 1636	MUTEX_EXIST("_x_X_UPDATE_X_x_") != 183
	b5085d1c9b420e3 : 1636	MUTEX_EXIST("***MUTEX***_PERSIST") != 183
	b5085d1c9b420e3 : 1636	HANDLE_OF_FILEPATH(\\.\NTICE) == 4294967295
✓	b5085d1c9b420e3 : 1636	MUTEX_EXIST("_x_X_PASSWORDLIST_X_x_") != 183
	b5085d1c9b420e3 : 1636	MUTEX_EXIST("_x_X_BLOCKMOUSE_X_x_") != 183
	b5085d1c9b420e3 : 1636	MUTEX_EXIST("***MUTEX***") != 183
	b5085d1c9b420e3 : 1636	HANDLE_OF_FILEPATH(\\.\SICE) == 4294967295

< 1 > 到第 1 页 确定 共 7 条 10 条/页

重新配置环境并上传

主动构造环境后行为

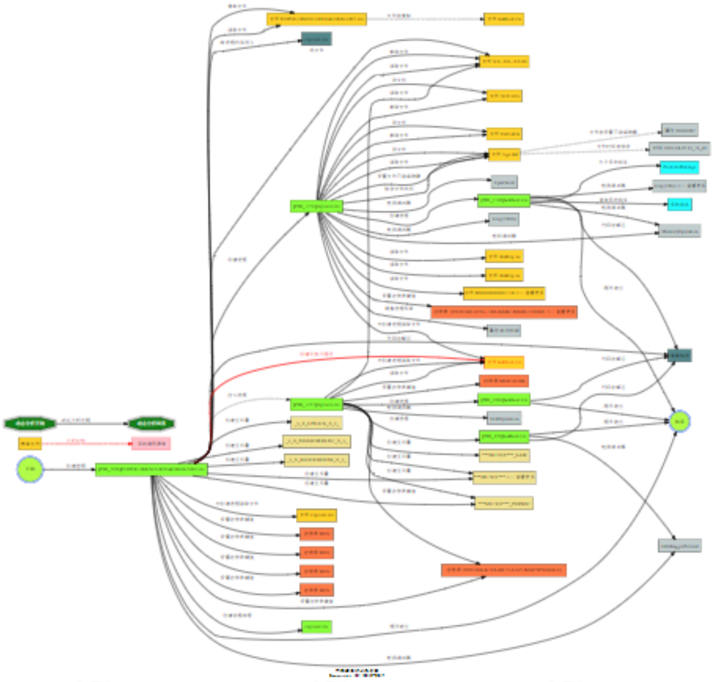
	检测调试环境	[普通]	检测调试环境，逃避分析与查杀
	删改注册表键	[普通]	删改注册表键 \REGISTRY\MACHINE\SOFTWARE\MICROSOFT\PCHEALTH\ERRORREPORTING\DW
+	删改注册表键值	[普通]	删改注册表键 \REGISTRY\MACHINE\SOFTWARE\MICROSOFT\PCHEALTH\ERRORREPORTING\DW 的值 D
	遍历模块列表	[敏感]	遍历系统中的模块
+	写入崩溃日志	[敏感]	程序崩溃，写入日志到 HarddiskVolume1\DOCUME~1\xpuser\LOCALS~1\Temp\4edb_appcompat.txt
	崩溃诊断	[敏感]	程序崩溃导致系统自动诊断
	收集崩溃信息	[普通]	从 explorer.exe 进程中收集崩溃信息
	诊断系统线程	[敏感]	诊断系统中的线程
	诊断线程	[敏感]	挂起 explorer.exe 进程中的线程，诊断崩溃原因
	诊断系统模块	[敏感]	诊断系统中的模块
+	创建转储文件	[敏感]	程序崩溃，创建dump文件 HarddiskVolume1\DOCUME~1\xpuser\LOCALS~1\Temp\77FCD.dmp
	访问系统服务	[敏感]	访问系统服务
	访问网络连接	[敏感]	通过 RASMAN 服务创建网络连接
	访问网络服务	[高危]	获取系统提供的socket服务



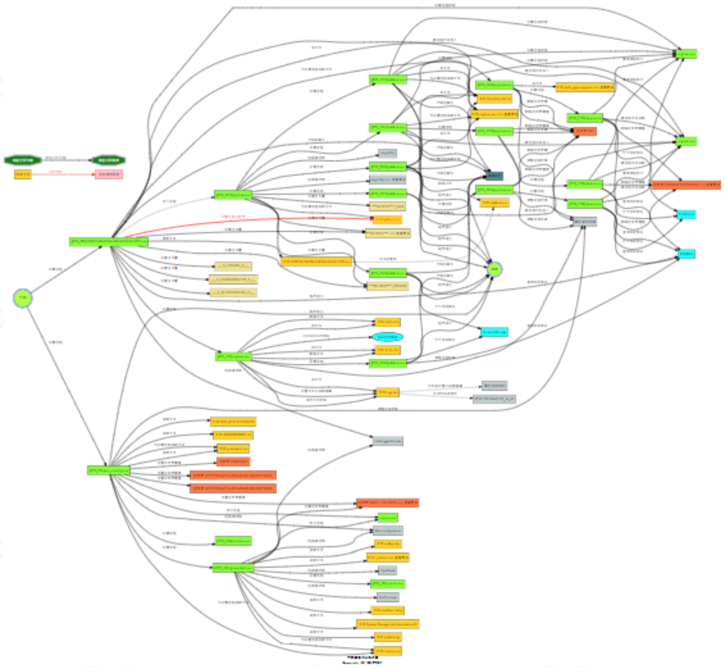
某恶意软件的执行环境构造实例

---------- 依据不同名字互斥量的创建情况分别实施不同的行为

原始报告行为



主动构造环境后行为



总结

一

沙箱是未来检测高级别攻击的重要手段，沙箱的检测与反检测对抗是未来攻防双方一项长期的“技术对抗”

二

该“技术对抗”是以实战为基础的高技术对抗，一方面需要紧密的结合实战动态与需求，另一方面也必须引入高技术手段

三

推动该技术的研究与进步需要业界紧密结合，期待产学研用紧密结合，共同推动相关技术进步，提升高级别威胁的防御能力

敬请批评指正！
欢迎加入团队！
谢谢！

苏璞睿
中国科学院软件研究所
邮件：purui@iscas.ac.cn